RNS-87-003

System Wide Performance Measurement for the Cray-2

William T.C. Kramer

Manager, High Speed Processors
Computational Services Branch
NAS Systems Division
NASA Ames Research Center
Moffett Field, CA 95014
(415) 694-4418

Much attention has been paid to individual program performance on supercomputers in the past. This paper presents issues and techniques for analyzing system wide performance and throughput for UNICOS on the Cray-2. A mixture of system utilities and custom software will be discussed and examples shown as to what parameters are important to maximize system throughput.

Introduction

One common justification for acquiring supercomputers are problems which can not done in a reasonable amount of time on any normal computer. This has led to many studies and articles emphasizing the performance of individual programs on supercomputers. Numerous articles have contributed to improving methods for coding algorithms and programs in order to achieve impressive performance on these machines.

However, most of the results do little to develop methods understanding overall system-wide performance for supercomputers. With the increase in interactivity of supercomputers afforded by UNICOS, maximizing single program performance is no longer sufficient to ensure that the expensive and scarce resource of the supercomputer is utilized to its maximum potential.

Traditional supercomputer operating systems have been improved to maximize performance for the batch environment. Much work has also been done for traditional UNIX™ systems to improve the system performance for interactive computing. With UNICOS it is possible and, indeed necessary, to combine these two approaches to obtain the best performance possible for the entire supercomputer system.

The following is a discussion of techniques used on the Cray-2 as NAS to increase the system throughput. Essentially, this paper describes approaches for solving the problems faced in managing a supercomputer facility using a philosophy normally associated with standard UNIX™ systems. That philosophy is one of building small simple tools and combining the tools together in the most effective manner.

The NAS System

The Numerical Aerodynamic Simulation (NAS) system is designed to provide one of the most advanced supercomputing environments for a national user base of scientists working primarily in the field of computational fluid dynamics and related disciplines. The goals of the NAS system are:

> *"1) to provide a national computational capability available to NASA, the Department of Defense (DOD) and other government agencies, industry, and universities, as a necessary element in ensuring continuing leadership in computational fluid dynamics and related disciplines;*
>
> *2) to act as a pathfinder in advanced, large-scale, computer systems capability through systematic incorporation of state-of-the-art improvements in computer hardware and software technologies; and*
>
> *3) to provide a strong research tool for NASA's Office of Aeronautics and Space Technology."* [1]

The NAS system began operation in September 1985, with the arrival of the first 256 Megaword (MW) Cray-2. The supercomputer is combined with a number of other systems, including numerous Silicon Graphics IRIS 2500T and 3030 workstations, Amdahl 5860 systems, and VAX 11/780 systems. All the systems are connected using Network Systems Corporation's Hyperchannel and all but the Cray-2 have ethernet connections.

A design requirement of the NAS system was to have a single, consistent user interface, regardless of the underlying hardware. The Cray-2 runs the UNICOS oerrating system, the Amdahls run Amdhal's UTS operating system, the Vaxes run the 4.3 Berkeley Distribution of UNIX™ and the IRIS workstations run GL/2-W3.5. All these varieties of UNIX™ are based on AT&T System V[2] with the addition of TCP/IP networking[3] and Berkeley UNIX extensions such as sockets[4]. The NAS Cray-2 was the first to UNICOS.

The goal of pathfinding in supercomputing dictates there be a minimum of vendor dependent solutions since NAS plans to always continue to move to the most advanced machine as quickly as possible. The use of UNIX™ across all our computing systems and the TCP/IP network allows great flexibility. For example, NQS, the Network Queuing System developed as NAS and now adapted by Cray and other vendors is used across all system instead of Cray Station. This approach makes management of supercomputers even more challenging.

Another implication of the use of UNIX™ and standard networking is the high level of interactive use of the Cray-2, particularly for interactive graphical analysis. The implementation of Silicon Graphics Remote Graphics Library on the Cray-2[5] allows the implementation of programs such as the Real-Time Interactive Particle Tracer, *rip*[6]. *Rip* is a distributed program that does graphics on the IRIS workstations and the computational fluid dynamics calculations on the Cray-2.

The Cray-2 at NAS is named *Navier* after the 18th century mathematician and fluid dynamist, Louis M.H. Navier. It is a fully configured system with 256 Megaword of 120 nano-second memory, 34 IBIS DD49 disk drives for a total of 40.8 Gigabytes (GB), and four Hyperchannel connections. As of this writing, a Cray-Tape-Channel connection is installed and being placed into production and a VME/ethernet interface is being delivered. The operating system is currently a field test version of UNICOS 3.0, which was placed into production on July 1, 1987. However, much of the work described in this paper was done prior to the arrival of UNICOS 3.0, which has improved functions for gathering performance information.

Methods of Measuring Performance

A number of performance measurement techniques may be used to monitor system efficiency. These techniques may be classified in at least three categories; benchmarking, usage analysis and performance monitoring. Techniques in all three categories are used at NAS and will be commented on, with the focus of this discussion on the latter two.

Benchmarking

Benchmarking is a method of evaluating supercomputer systems during which individual programs and collections of programs are run and their performance analyzed. The program performance is compared to a pre-determined baseline performance. The comparison yields the relative performance of a system or program compared to the baseline.

Benchmarking is an important activity at NAS. A suite of benchmarking codes have been developed and continue to be refined to reflect the work load. These benchmarks have been used in acquisitions of computer equipment and for changes in hardware and software. The suites combine commonly used application programs with typical data sets such as the NAS kernel algorithms which are utilities in use by many of the at NAS programs.[7] Other benchmarks apply to system and network performance[8].

With respect to system management, the benchmark suites are used for several purposes. A set of shell scripts have been developed to facilitate running these benchmarks so the entire suite or subsets of it may be run quickly and easily. The scripts run the benchmark programs, evaluate performance and record the results. In this way, it is possible to evaluate changes in the system, whether it be a new version of a compiler or a complete new computer system.

Before any significant change in the production system or compilers is made, the benchmarking suites are run with the new system and results give an expectation of changes in performance that may be expected as well as making it possible to identify potential problems.

In a similar procedure, a list of known bugs and the test codes demonstrating these problems is maintained. When new releases of products are provided, these test cases are run and a measure of the improved reliability is obtained. Benchmarking allows a general feeling of confidence when a new release of the system software is put into production.

Usage Analysis

One of the most effective performance measurement procedures at NAS is the analysis of the system usage. It represents a long term picture of system performance as well as providing insight into the how clients are using the system. System usage data has several advantages at NAS. The data is generally available from UNICOS, and is in a format similar to other UNIX™ implementations. In particular, system usage data is recorded in the file /usr/adm/pacct whenever a process terminates[9] and further usage information is stored in other files.

Reporting Usage

When the Cray-2 was first delivered to the NAS, there was a clear need to record, report and monitor the usage of the system. While no money is charged for NAS usage, individual projects and classes of users are allocated Cray-2 CPU time through a peer or management review process. It was necessary to track and accurately record the usage of system by individual, project, and class.

Journal accounting software, *ja*, from AIM Technology, Inc was first used to post process and record standard UNIX™ accounting data. This software was extensively modified by the NAS staff for running the Cray-2, to improve its function and performance, tailor it to the needs of NAS and incorporate new data and features available in newer releases of UNICOS. Currently, *ja* records and reports 16 categories of usage for each user every day. The categories include CPU time, memory, I/O transfers and requests and is separated into prime and non-prime time. CPU time is further separated into three categories for interactive, batch and deferred batch, and is broken down for multitasking usage. Users and project leaders can query the usage on-line through a single command. An example of a request is shown in Figure 1.

A number of usage reports are generated weekly and monthly. Management gets a weekly summary of usage for each user and project, including cumulative summaries. Figure 2 shows the summary table for a single week by major user classes. In this example, the interactive use (defined to be processes with nice values of 24 or less) is shown as approximately 11%, and 17% during prime time. This is slightly lower than the overall yearly average of 15 to 20% of interactive usage, when clients were developing their codes. Comparisons over time of summary numbers such as these point to shifts in the usage profile and should lead to shifting system management policies.

NQS Usage

Obviously, NQS batch processing is important to NAS, a situation typical of supercomputing but not typical of standard UNIX™ environments. There are three major queue structures for the Cray-2; normal queues, deferred queues and a special queue. The normal queue, is a piped queue for 8 client queues with different limits. Jobs entering the pipe queue *navier* on either the Cray-2 or any major network nodes are feed into one of the 8 client queues depending on the limits the submitter specified. Jobs are run as soon as possible. The defer queue is a pipe queue for 2 client queues. Jobs in the defer queue are run only when there is not enough work in the normal queues to keep the system busy. Different charging rates are used since it may be several days before a job is executed. The special queue is used for jobs requiring special treatment. This queue is used for very large and long running jobs. Figure 3 is a table summarizing the NQS parameters currently in use at NAS.

Modifications to NQS were made to record the queue from which each process runs. It then becomes possible to record and report utilization by queue, as shown in Figure 4. This table provides a calculated value which is indicative of the time jobs wait in the queue before starting compared to the time it took to run jobs.

Ad Hoc Usage Analysis

Very often, questions arise which are not answered by defined reports and charts, but which do not require the creation of new reports. Easy access to and flexible analysis tools for accounting data allow these questions to be quickly addressed.

An example of this type of analysis occurred when the UNICOS 3.0 field test starting in June, 1987. The question was whether to allow the use on the Cray-2 of full screen editors such as *vi* and a micro version of *emacs*, since the addition overhead of character mode I/O was unknown. The suspect programs were placed on the system and over a two week period their usage was observed. Figure 5 shows the results of the analysis. For each editor, the ratio of CPU time to I/O was compared to the basic editor, *ed*. The results show that editors with improved functionality have a better CPU to I/O ratio than on support systems. When the additional cost of transferring files to and from the other systems which allowed full screen editing was considered, the decision to provide full screen editors on the Cray-2 was made. It is now up to our users to balance the additional CPU cost, charged against their allocations, with the added function.

Usage Analysis Summary

The major point of this section is a combination of a programs, such as *ja* and a number of shell scripts, such as those that extract the data from *ja* and format it into the tables using UNIX™ utilities such as *tbl* and *troff*, simplify the recording and reporting of usage data. Very few system modifications were made in order to extract the data, and possibly more important, the techniques and procedures used are portable to other UNIX™ systems.

Nonetheless, usage analysis has some limitations. It does not allow a true picture of the state of the system at any moment in time. Typically, most supercomputer applications execute for a long time, in cases several hours. Since UNIX™ usage data is recorded only when a process terminates, it is not possible to deduce what the system is doing at any particular moment. This prevents the usage data from being used to tune system performance parameters directly.

Performance Monitoring

As noted above, usage analysis is not sufficient for providing a complete picture. Real-time performance monitoring is also necessary for managing a system to provide the best service. The basic structure of these procedures is to execute at set time intervals, typically 5 or 10 minutes, gather statistics, record a summation of statistics and possibly modify system parameters. Several examples of this type of monitoring are provided.

Idle Time and Memory Usage

One of the major measures of system performance used at NAS is the amount of idle time the system accumulates. The "nice value" priority scheduling of UNICOS allows idle time to be kept very low while system response to interactive jobs remains good. There is always some idle accumulated when the system is started, before all the user level codes are started. In general, idle time is kept below 3% on a long term average and less than 1% on a short term basis. On the other hand, system performance, and particularly larger interactive graphics programs, are very sensitive to swapping. Since the very large processes do not swap unless they are expanding in memory, anytime swapping occurs impacts the interactive use of the system. A balance is needed to give the proper performance to all users.

Initially, it was necessary to understand when idle time occurred and how it related to system responsiveness. Memory usage needed to be monitored evaluate when the use of too much memory degraded the service. Examples of the graphs reporting the idle time and memory usage used for this analysis are shown in Figures 6 and 7.

Another piece of the puzzle, shown in Figure 8, was to measure the load on the system. The load average gives an indication of system responsiveness by showing how many processes are competing for the CPU resource. Load average is a UNIX™ measurement which is a moving average of the number of processes in the kernel's run queue[10] (the kernel has several internal queues which are distinct from NQS queues.) There were also periodic tests of network transfer rates to judge performance. All these items were reviewed as system management parameters were changed, thus providing a check to insure the changes improving the system.

NQS Queue Control

Ensuring the Cray-2 is utilized to the maximum degree requires constant monitoring and modification of parameters. The trade off between maximum CPU utilization, system wide throughput and interactive response for real-time graphics required automating functions normally given to operators of the system. With all other concerns, human operators generally do not have the time to quickly respond to changes in system behavior, particularly since they do not have control over scheduling the interactive usage of the system.

The decision was made to automate the process of controlling how many and which batch jobs are be allowed to run on the system as much as possible. Furthermore it was decided to automate through the use of a shell script instead of modifying NQS and other system utilities. This decision had several distinct advantages:

1) It did not require maintaining source code modifications to CRI products, and facilitates moving to new versions of software (as shown by by moving UNICOS 3.0 within a week of arrival);

2) The development time is very short;

3) Procedures can be maintained and modified by a system administrator instead of a system programmer; and

4) The solutions are portable to other systems if necessary.

The script monitors the idle time and memory usage of the system at frequent intervals. Threshold limits for idle time and memory, based on time of day, have been developed over the past several months. Idle time is calculated by comparing the accumulated CPU time of the 4 idle processes with the idle time accumulated at the last interval. If idle time is above the threshold level, the script will increase the run limits on some of the normal NQS queues. If there are no jobs in the normal queues, the script then starts jobs in the defer queue.

Memory is also a consideration for the script. Initially, jobs were started regardless of the idle time if the amount of memory in use was below a relatively high limit. This approach was modified since the number of jobs in the system become so many that they were no longer completing in a reasonable time and would adversely affect interactive usage. The memory limits are now adjusted so that while the processors remain totally busy, the memory average is about 150MW instead of 225 MW.

The script is undergoing continued improvement, and other system management scripts like this have been added. A script to monitor users stacking and running many jobs in a queue at one time has enabled us to fairly and consistently enforce the system policy which limits the number of jobs a user can have run in any queue with minimum effort.

Sar

UNICOS 3.0 provides a Cray-2 implementation of the standard UNIX™ utility *sar* for system activity reporting. Typically, *sar* is scheduled to run periodically, generally with a *crontab* entry and record system behavior which later can be reported. Initial usage of *sar* indicates it is a useful and accurate tool. Results agree with data obtained using locally developed scripts. For example, recording idle time by tracking the idle processes CPU time and with *sar* indicates a good match between *sar* and our previous methods of recording the same information. *Sar* also reports data which was not easily obtained previously, such as the amount of CPU time spent in I/O wait and in system overhead and statistics for individual devices. In all, *sar* should improve the monitoring of the system.

Monitoring Summary

It is necessary to monitor an interactive supercomputer frequently and to modify parameters to maximize performance. Fortunately, much of this can be often done with shell scripts, whose creation and maintenance is generally simpler than programs and modifications of system utilities.

Conclusions

Managing a supercomputer to get the maximum utilization has always been a difficult task. The additional challenges presented by the Cray-2 and the interactivity of UNICOS further complicate the task. At NAS, we have been dealing with these problems for more than a year while supporting a national user community.

A wide range of tools are used to measure system performance. These tools include benchmarks which measure performance of specific programs and in specific environments in order to compare performance in a formal manner. Collection and analysis of various system usage data is another valuable component. Usage data gives insight into particular usage, and for long term trends. It has also been used to develop policy and determine the initial values for system parameters. Finally, it is necessary to monitor system behavior and performance, frequently - on the order of every few minutes - to detect short term changes in system usage and to modify parameters.

Fortunately, UNICOS provides a great deal of flexibility to accomplish these tasks, making traditional approaches of complex scheduling algorithms and system modifications for the most part unnecessary. The UNIX™ philosophy of building small simple tools and using shell scripts to combine those tools can be adapted to effectively manage supercomputers. This results in implementing changes more quickly, providing a better responsiveness and more easily maintaining changes.

Acknowledgments

It is necessary to point out that the work described here is the combined contribution of many people, working for the NASA, General Electric (the Integration Support Contractor), Sterling Software (the System Support Contractor) and Cray Research. Not to go unmentioned are the users of NAS, who have contributed new and unique ideas to our system management procedures.

[1] Bailey, F. Ron: *Status and Projections of the NAS Program.* **Proceedings of a Workshop on Supercomputing Environments**, NASA-Ames Research Center, Moffett Field, CA, June 24-26, 1986.

[2] Kevorkian, D.E., ed.: **System V Interface Definition.** Indianapolis, IN, 1985.

[3] Feinler, E.J., et.al., eds: **DDN Protocol Handbook.** Vol 1, Defense Technical Information Center, Alexandria, VA 1985.

[4] Laffler, S.J.: **A 4.2BSD Interprocess Communication Primer.** Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 1983.

[5] Choi, Diana and Levit, Creon: *An Implementation of a Distributed Interactive Graphics System in an Supercomputer Environment*, **Proceedings of the Second International Conference on Supercomputing,** Vol 2, International Supercomputing Institute, Inc. May, 1987.

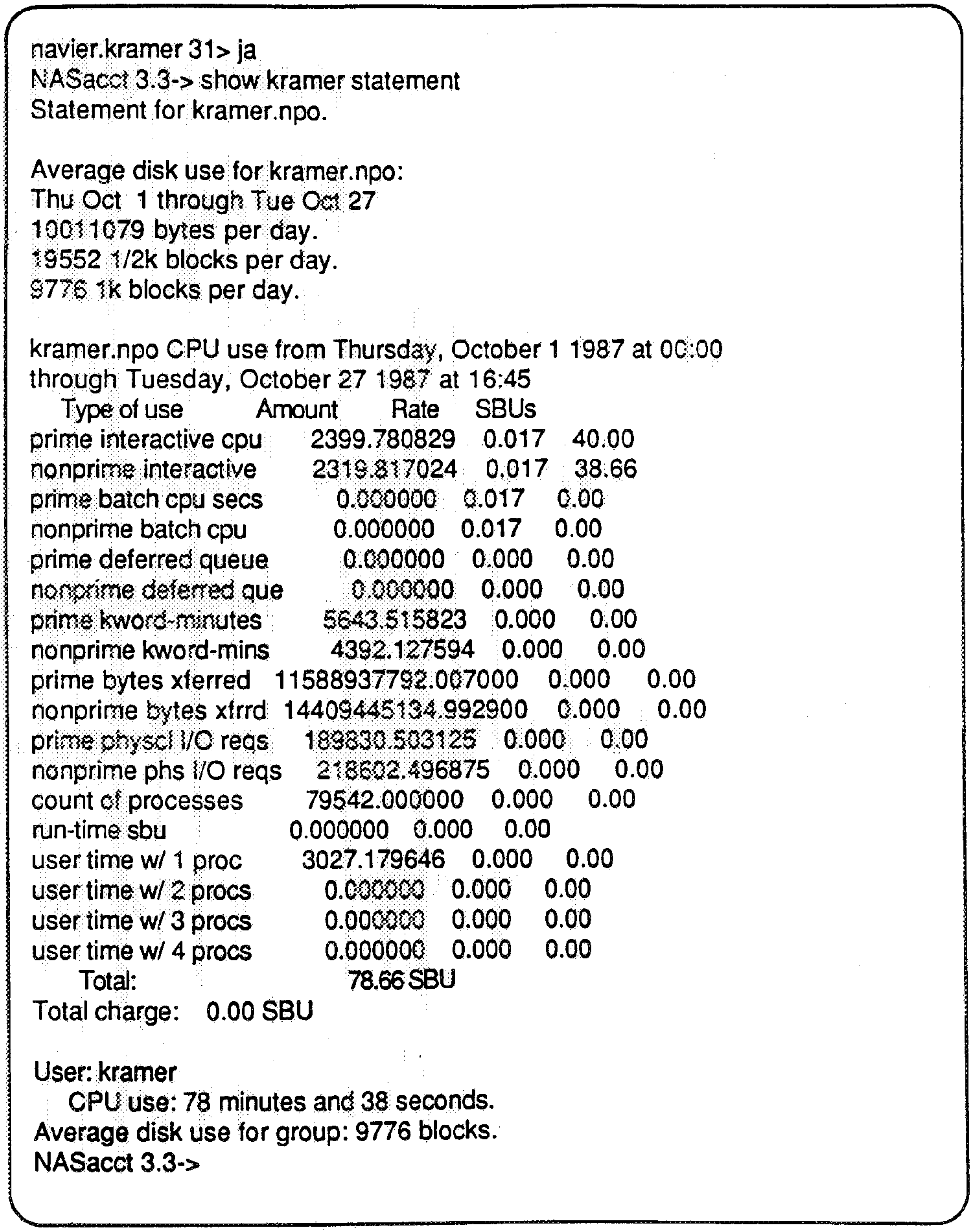
[6] Rogers, S, Merritt, F and Choi, D.: *RIP(Real-Time Interactive Particle Tracer).* A computer program. NASA Ames Research Center, Moffett Field, CA 1986.

[7] Bailey, David H.: *NAS Kernel Benchmark Results.* **First International Conference on Supercomputing Systems,** IEEE Computer Society, 1985, pp 341-345.

[8] **Initial and High Speed Processor 2 (HSP 2) Computer System Request for Proposal.** NASA Ames Research Center, Moffett Field, CA April 1987.

[9] **UNICOS System Administrator Guide for Cray-2 Computer Systems.** SG-2019, Cray Research, Inc., 1987.

[10] Bach, Maurice J.:**The Design of the UNIX Operating System.**, Prentice-Hall, Inc., 1986.



```
navier.kramer 31> ja
NASacct 3.3-> show kramer statement
Statement for kramer.npo.

Average disk use for kramer.npo:
Thu Oct  1 through Tue Oct 27
10011079 bytes per day.
19552 1/2k blocks per day.
9776 1k blocks per day.

kramer.npo CPU use from Thursday, October 1 1987 at 00:00
through Tuesday, October 27 1987 at 16:45
   Type of use         Amount       Rate    SBUs
prime interactive cpu      2399.780829   0.017   40.00
nonprime interactive       2319.817024   0.017   38.66
prime batch cpu secs         0.000000   0.017    0.00
nonprime batch cpu           0.000000   0.017    0.00
prime deferred queue          0.000000   0.000    0.00
nonprime deferred que          0.000000   0.000    0.00
prime kword-minutes        5643.515823   0.000    0.00
nonprime kword-mins        4392.127594   0.000    0.00
prime bytes xferred   11588937792.007000   0.000    0.00
nonprime bytes xfrrd  14409445134.992900   0.000    0.00
prime physcl I/O reqs    189830.503125   0.000    0.00
nonprime phs I/O reqs     218602.496875   0.000    0.00
count of processes       79542.000000   0.000    0.00
run-time sbu            0.000000   0.000    0.00
user time w/ 1 proc      3027.179646   0.000    0.00
user time w/ 2 procs        0.000000   0.000    0.00
user time w/ 3 procs        0.000000   0.000    0.00
user time w/ 4 procs        0.000000   0.000    0.00
     Total:                  78.66 SBU
Total charge:   0.00 SBU

User: kramer
   CPU use: 78 minutes and 38 seconds.
Average disk use for group: 9776 blocks.
NASacct 3.3->
```

Figure 1

Sample output from the accounting program ja, showing a complete list of usage for the current month

Navier Resource Use Breakdown

Week Ending 09/27/87

Class		**Interact.** *minutes*	**Batch** *minutes*	**dfrrd-q** *minutes*	**cpu** *total*	**ave size** *core MW*	**I/O** *MWords*	*k reqs*
NASA_ARC	**prime**	1110	3681	77	4868	3.159	15040	2156
	nonprm	295	5179	1372	6846	9.848	20429	1200
NASA_LaRC	**prime**	1028	5121	37	6186	1.499	3281	1815
	nonprm	54	2184	974	3212	7.022	670	441
NASA_LeRC	**prime**	6	1404	0	1410	10.171	320	51
	nonprm	1	661	78	740	11.188	78	12
Other_NASA_	**prime**	3	31	0	34	12.987	40	15
	nonprm	0	42	0	42	17.378	19	2
DOD	**prime**	92	1752	177	2022	2.139	1383	454
	nonprm	16	1162	2032	3211	3.428	500	117
Commercial	**prime**	109	846	0	955	3.249	512	18118
	nonprm	82	1071	0	1154	6.527	189	19305
University	**prime**	41	279	13	334	4.286	1977	175
	nonprm	43	417	1924	2384	7.136	2946	207
PRGSUP	**prime**	253	312	0	565	0.216	2281	1142
	nonprm	46	1	10	57	0.114	903	182
OVERHEAD	**prime**	163	23	0	186	0.099	1135	664
	nonprm	438	3	0	440	0.113	8463	4897
total	*prime*	*2806*	*13448*	*305*	*16559*	*2.926*	*25967*	*24590*
	nonprm	*974*	*10721*	*6390*	*18085*	*7.443*	*34195*	*26365*
	total	*3781*	*24169*	*6696*	*34645*	*5.284*	*60163*	*50955*

Figure 2

Current NQS Queues

Queue	*Memory MW*	*CPU Seconds*	*Run Limit*	*Nice Value*	*Jobs/User*	*Queue Priority*
(Interactive)	20	600		20/24	2	
Short	16	1,200	8	25	2	10
Medium	16	3,600	9	27	1	10
Long	16	7,200	9	28	1	8
Large	40	7,200	5	27	*	8
Xlong	40	14,400	5	27	*	6
Smalldebug	12	100	10	24	1	14
Bigdebug	125	300	1	24	*	12
Big	125	14,400	#	26	*	6
Deferss	16	2,700	2	28	1	10
Deferll	100	5,400	2	28	1	10

* Only one job per user should be running at a given time in any *one* of these queues.

The *Big* queue will be active from 6 pm until 3 am PST. While it is active, the run limit will initially be set to 3.

Figure 3

Navier NQS Utilization Report

For week ending 09/28/87

	number	*cpu minutes*		*turn around*	*delay*
Queue name	*of jobs*	*total*	*median*	*median minutes*	*factor*
big	27	655	4.5	60.0	13.2
bigdebug	48	29	0.1	2.7	27.5
deferll	78	4238	72.7	286.3	3.9
deferss	162	3012	22.4	67.4	3.0
large	134	1989	2.2	80.0	36.3
long	143	4366	5.1	325.4	63.6
medium	251	4034	14.0	76.3	5.4
short	246	935	1.9	16.7	8.7
smalldebug	424	139	0.2	2.1	9.3
special	2	38	19.1	303.7	15.9
xlong	86	6220	9.4	481.1	51.3
total	*1601*	*25655*			

Note: Time for abnormally terminated processes not included in jobs.)

Figure 4

Editor	Cray-2	Vax 11/780
ed	1.0	1.0
ex	2.1	2.6
vi	3.3	4.5
emacs	31.8 *	16.3

Figure 5

A comparison of CPU time to I/O for several editors was made to determine the overhead of character I/O.

* A version of micro-emacs ported by a user

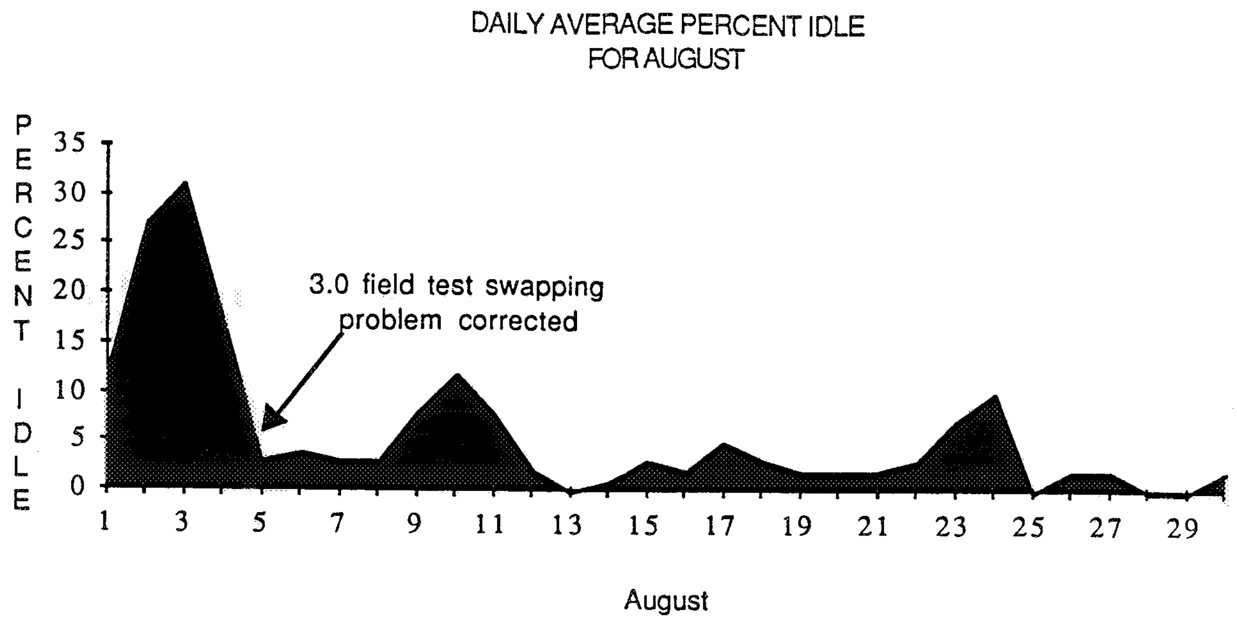


Figure 6

Daily average idle time for the month of August

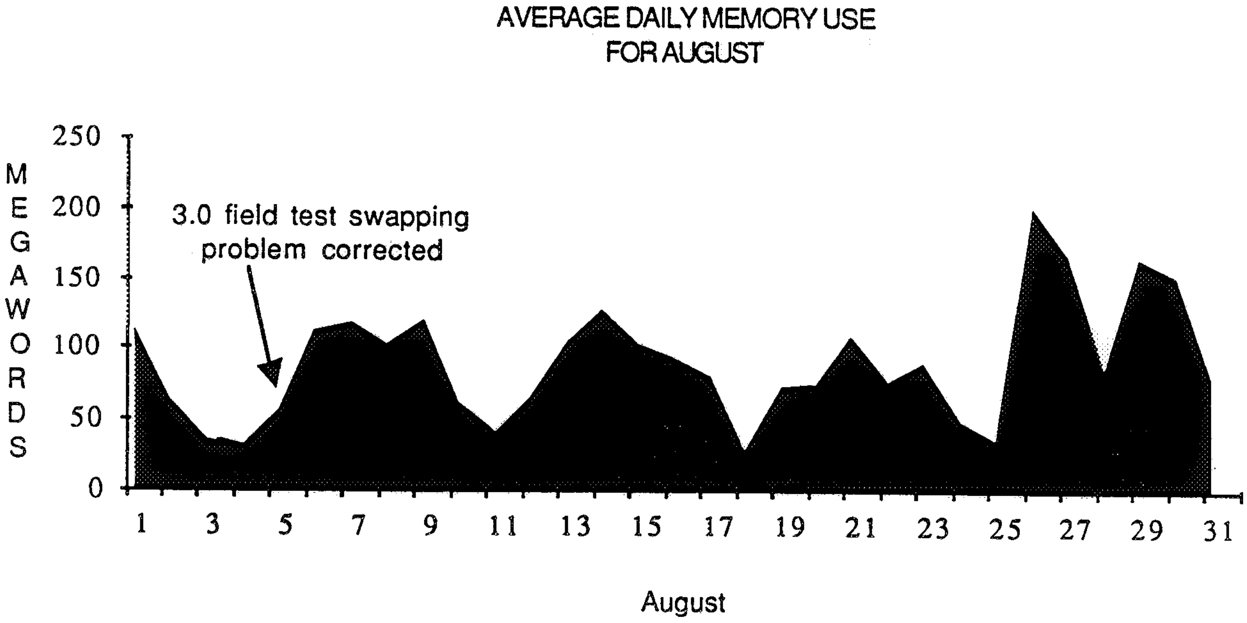


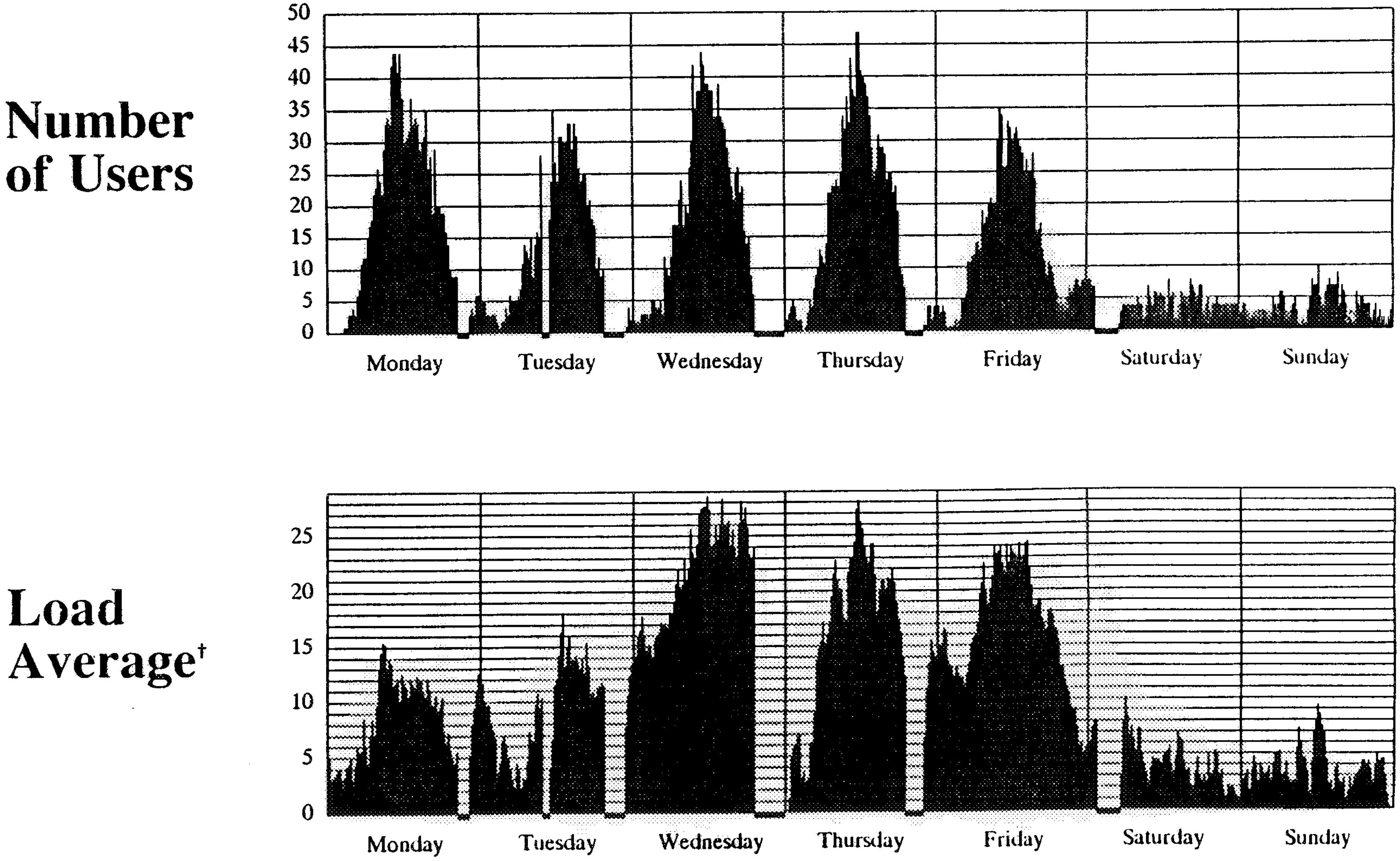
Figure 7

Daily average memory usage for the month of August

NAVIER

Oct 5 – Oct 11

Figure 8



†The number of processes competing for the CPU(s), averaged over 15 minutes